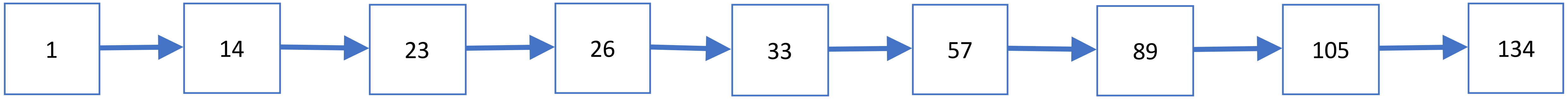
Skip Lists

Sorted Array

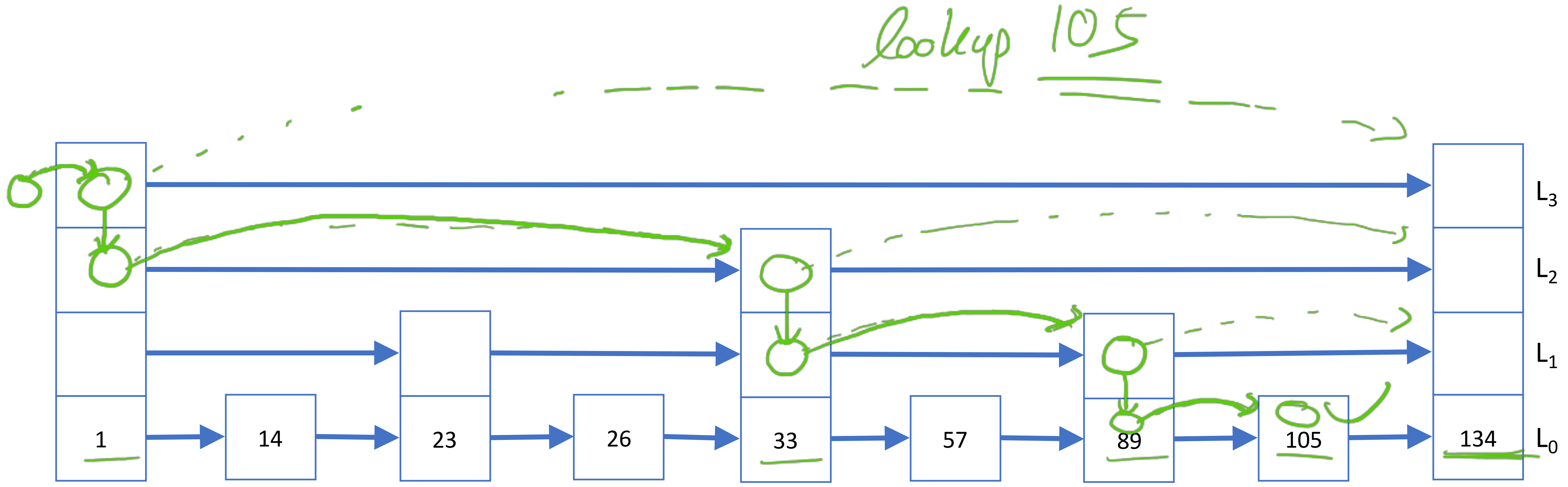
1	14	23	26	33	57	89	105	134

Sorted Linked List



Skip Lists: Main Idea

Introduce "express lanes" into linked list



Complexity of Lookup With Pre-Built Lanes

N: # of keys in list

Steps for lookup on every lane:

Const.

1) Peek ahead, then either:
   - 2a) Move to slower lane, or
   - 2b) Take 1 step ahead, then move to slower lane

With $\log_2(N)$ lanes: $O(\log(N))$

Problem: Insertion and Deletion

- Changing the base list invalidates the express lane structure

- Instead of rebuilding or repairing it, we
  - Commit to using an approximate express lane structure
  - Perform approximation step with each insertion

Skip List: Insertion Algorithm

1. Flip coin for each lane >0 from lowest to highest

   heads: new key is on lane

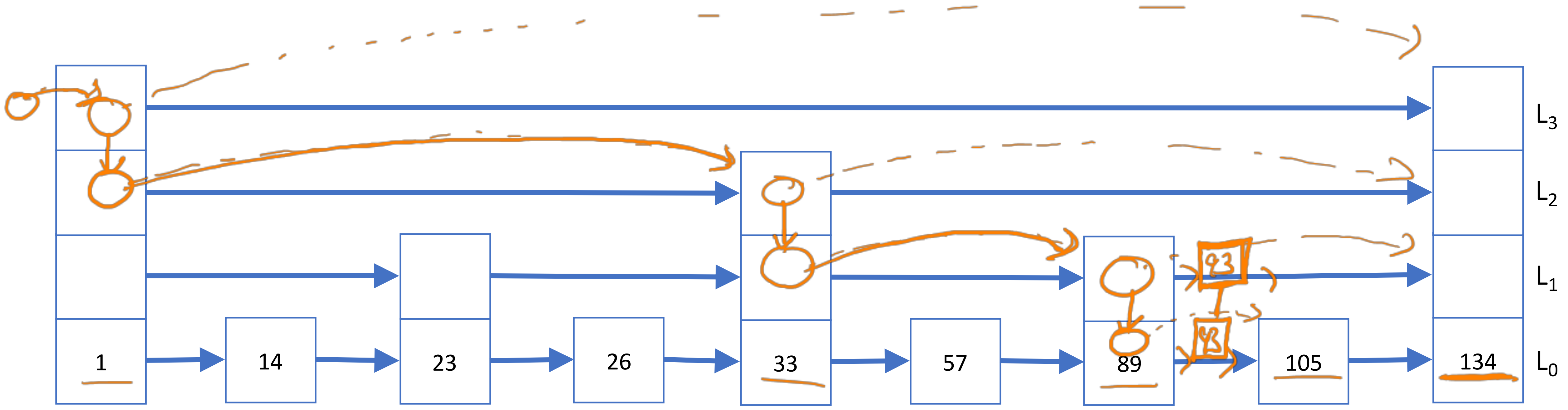
   tails: new key is not on lane, and we stop flipping

2. Perform lookup for new key and insert it into each of the lanes it is on

Insertion Example

Insert 93

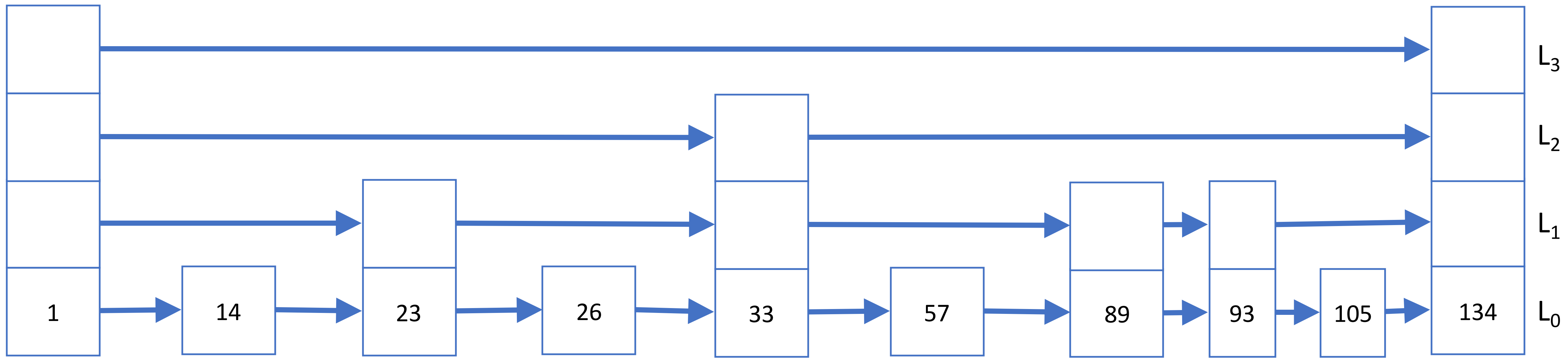
L_2	:	tails	25%
L_1	:	heads	50%
L_0			100%



Insertion Example

Insert 93

L_2	:	tails	25%
L_1	:	heads	50%
L_0			100%



Insertion Example

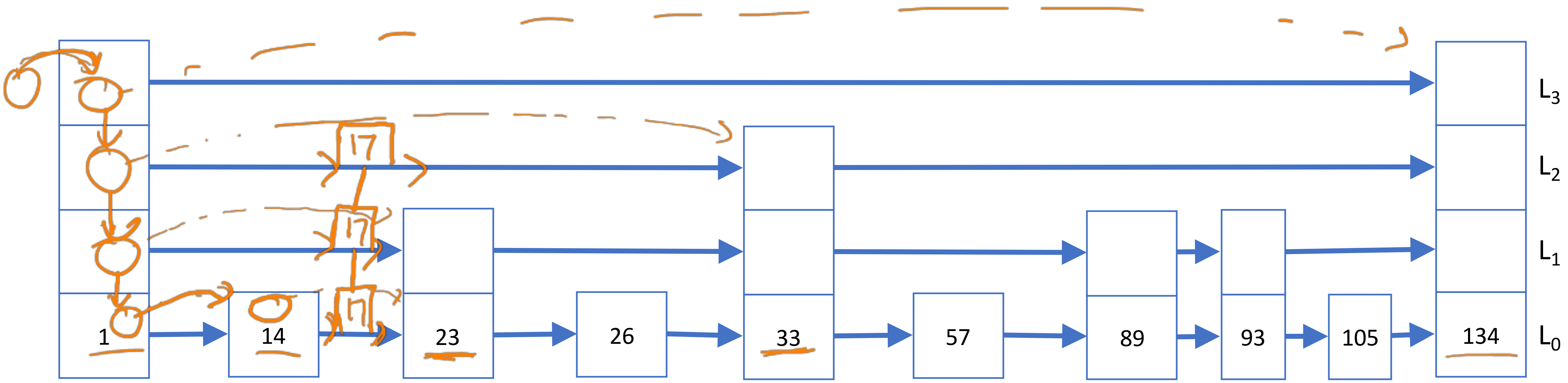
Insert 17

L_3 : tails

L_2 : heads

L_1 : heads

L_0



Insertion Example

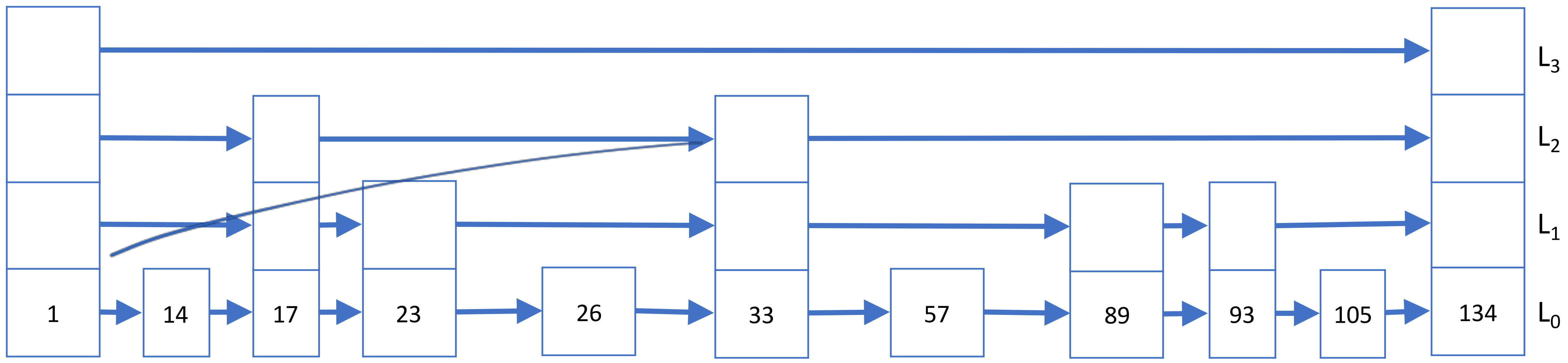
Insert 17 ✓

L_3 : tails

L_2 : heads

L_1 : heads

L_0



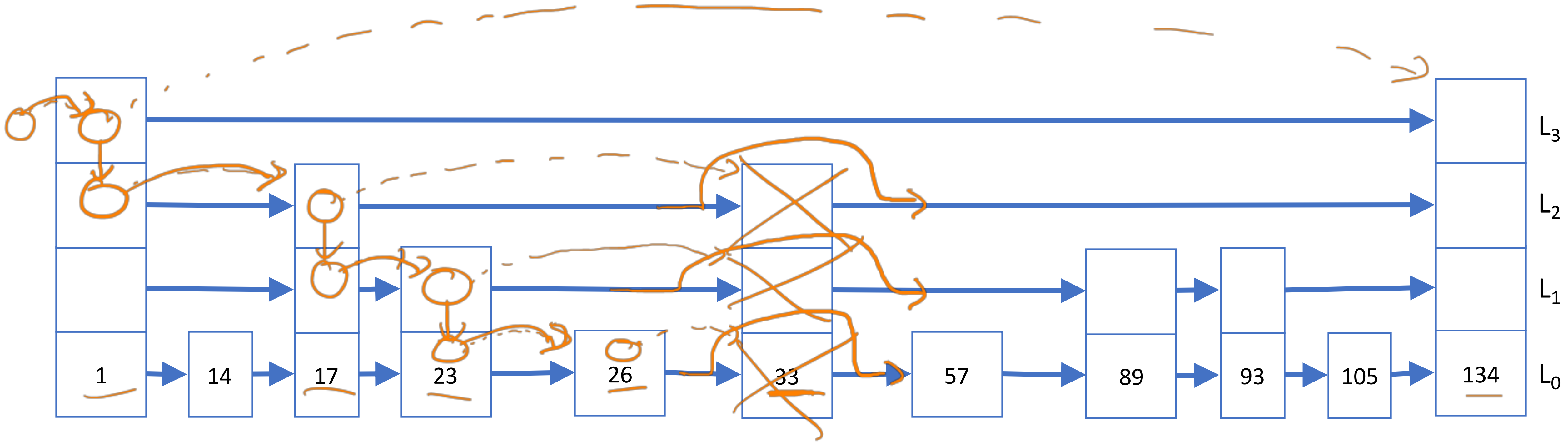
Skip List: Deletion

1. Perform lookup for key to delete, but don’t step into node containing key

2. Delete key from every lane it is on while continuing lookup procedure

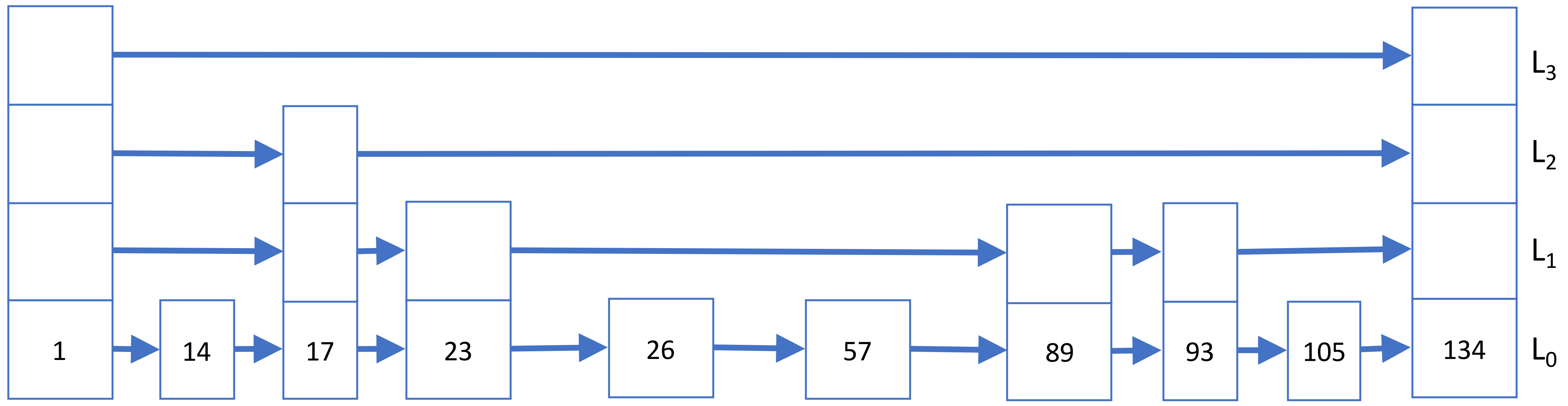
Deletion Example

Delete 33



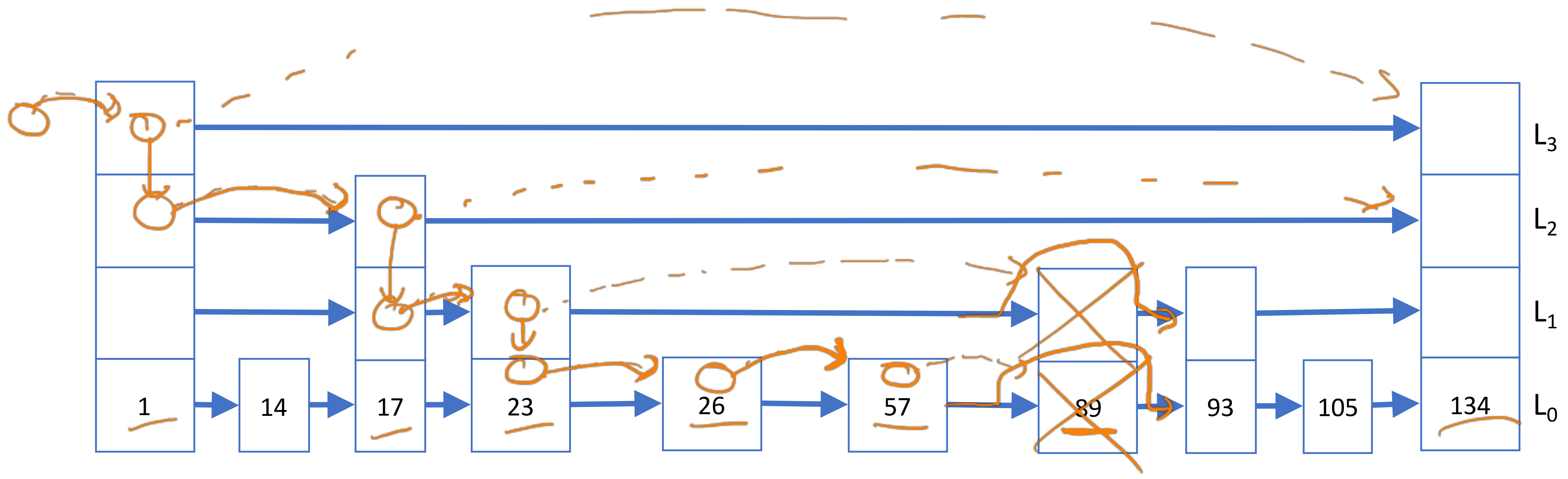
Deletion Example

Delete $\underline{33}$ ✓



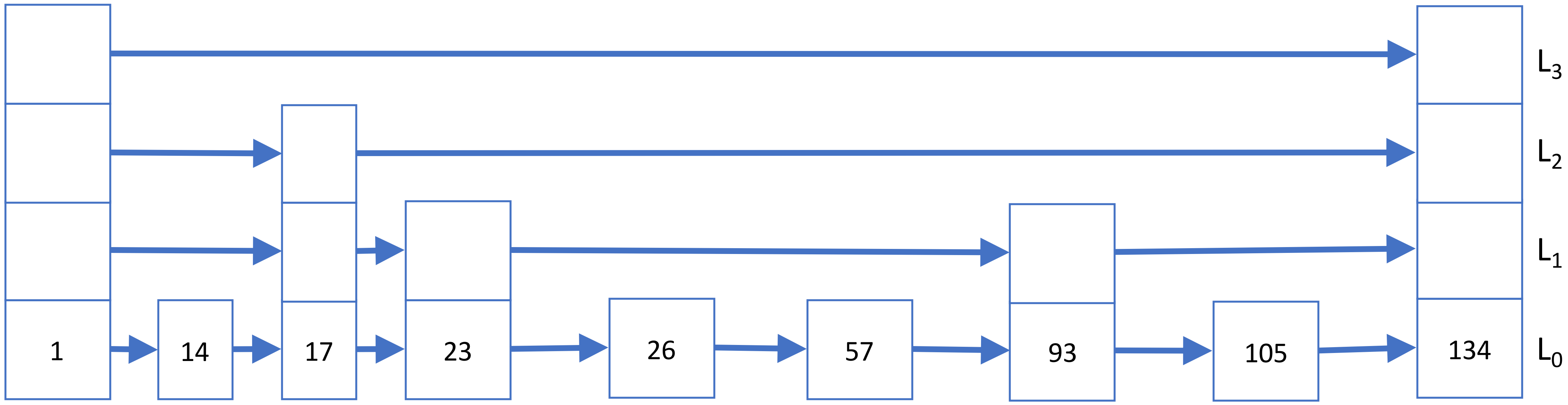
Deletion Example

Delete 89

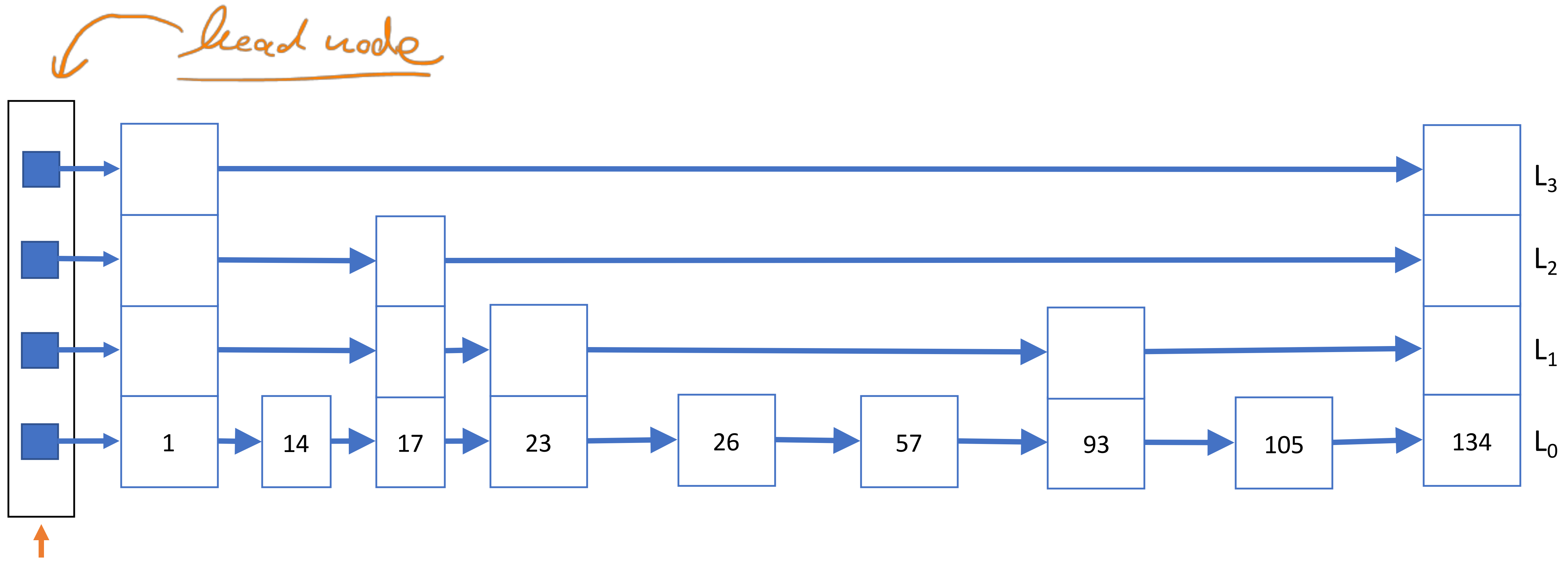


Deletion Example

Delete 89



Access to First Nodes



List for access to the first node of every lane